Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

Claim 1 (currently amended):

A program flow method in a program component system, the program component system comprising a running time system and several components each having ~~one~~ a program portion, said method comprising the following steps that are performed after a first component of said several components has been called and during the execution of the program portion of ~~a~~ said first component:

a) acquiring ~~data acquisition~~ for said first component, by means of the running time system, ~~of~~ data ~~of~~ from a second component of said several components ~~into said first component~~ without any need for programmer-defined interfaces in said second component, wherein the source of the data to be acquired within said second component is selected according to a definition of said first component; and

b) disposing, from said first component, ~~data disposal~~ by means of the running time system, data ~~of said first component~~ into said second component without any need for programmer-defined interfaces in said second component, wherein a target to which said data is to be deposited within said second component is selected according to a definition of the first component.

Claim 2 (currently amended):

The method according to claim 1, characterized in that the data transmitted during ~~the data acquisition~~ said acquiring are transferred from a memory image portion of said second component into a transfer data region of said first component, and/or that the data transmitted during ~~the data disposal~~ said disposing are transferred from a transfer data region of said first component into a memory image portion of said second component.

Claim 3 (currently amended):

The method according to claim 1, characterized in that said ~~data acquisition~~ acquiring and/or ~~data disposal~~ said disposing is carried out without the cooperation of said second component.

Claim 4 (currently amended):

The method according to claim 1, characterized in that said second component is inactive during said ~~data-acquisition~~ acquiring and/or ~~data-disposal~~ said disposing.

Claim 5 (currently amended):

The method according to claim ~~1~~ 2, characterized in that said transfer data region of said second component is located in a saving region during said ~~data-acquisition~~ acquiring and said ~~data-disposal~~ disposing.

Claim 6 (currently amended):

The method according to claim 1, characterized in that local and/or non-persistent data of said second component are transmitted during said ~~data-acquisition~~ acquiring and/or said ~~data-disposal~~ disposing.

Claim 7 (currently amended):

The method according to claim 1, characterized in that during the execution of the program portion of said first component a waiting list is made indicating which of the data of said first component require ~~data-disposal~~ disposing.

Claim 8 (currently amended):

The method according to claim 1, characterized in that a called component ~~an~~ can directly access an access data region comprising the data fields defined and /or available in ~~said~~ a calling component.

Claim 9 (currently amended):

The method according to claim 1, characterized in that ~~the~~ a call of a component is triggered by call information comprised in a docking point of ~~the~~ a calling component.

Claim 10 (currently amended):

A method of expanding a program component system comprising several components[~~,~~] by ~~one~~ a further component, said method comprising the steps of:

a) searching for docking points for said further component in said program component system, which docking points correspond to an inheritance parameter determined by a definition of said further component; and

b) modifying ~~the components~~ each of said several components of the program component system ~~where~~ in which at least one docking point was found[~~,~~] by entering call information ~~that indicates the further component~~ at each docking point found, said call information indicating said further component, wherein said expansion of said program component system is completed without any need for programmer-defined expansion interfaces in said several components.

Claim 11 (currently amended):

The method according to claim 10, characterized in that all interaction interfaces of ~~the previous~~ said several components are predefined as potential docking points.

Claim 12 (currently amended):

The method according to claim 10, characterized in that all interaction screen fields referenced by ~~the previous~~ said several components and/or all print mask output fields and/or all access operations on persistent data are predefined as potential docking points.

Claim 13 (currently amended):

The method according to claim 10, characterized in that ~~by~~ said entering ~~said~~ call information ~~into a~~ at each docking point found includes preparing a call of the further component from ~~the~~ each of said several components ~~component~~ into which said call information was entered~~, is prepared~~.

Claim 14 (currently amended):

The method according to claim 10, characterized by ~~an additional step of~~ additionally:

c) generating at least one binary object from the definition of the further component.

Claim 15 (original):

The method according to claim 14, characterized in that a maximum of one binary object is generated for each docking point that has been found.

Claim 16 (previously amended):

The method according to claim 15, characterized in that while generating each binary object, the memory allocation is considered in the one component of the program component system which includes the underlying docking point.